

Fig. 1

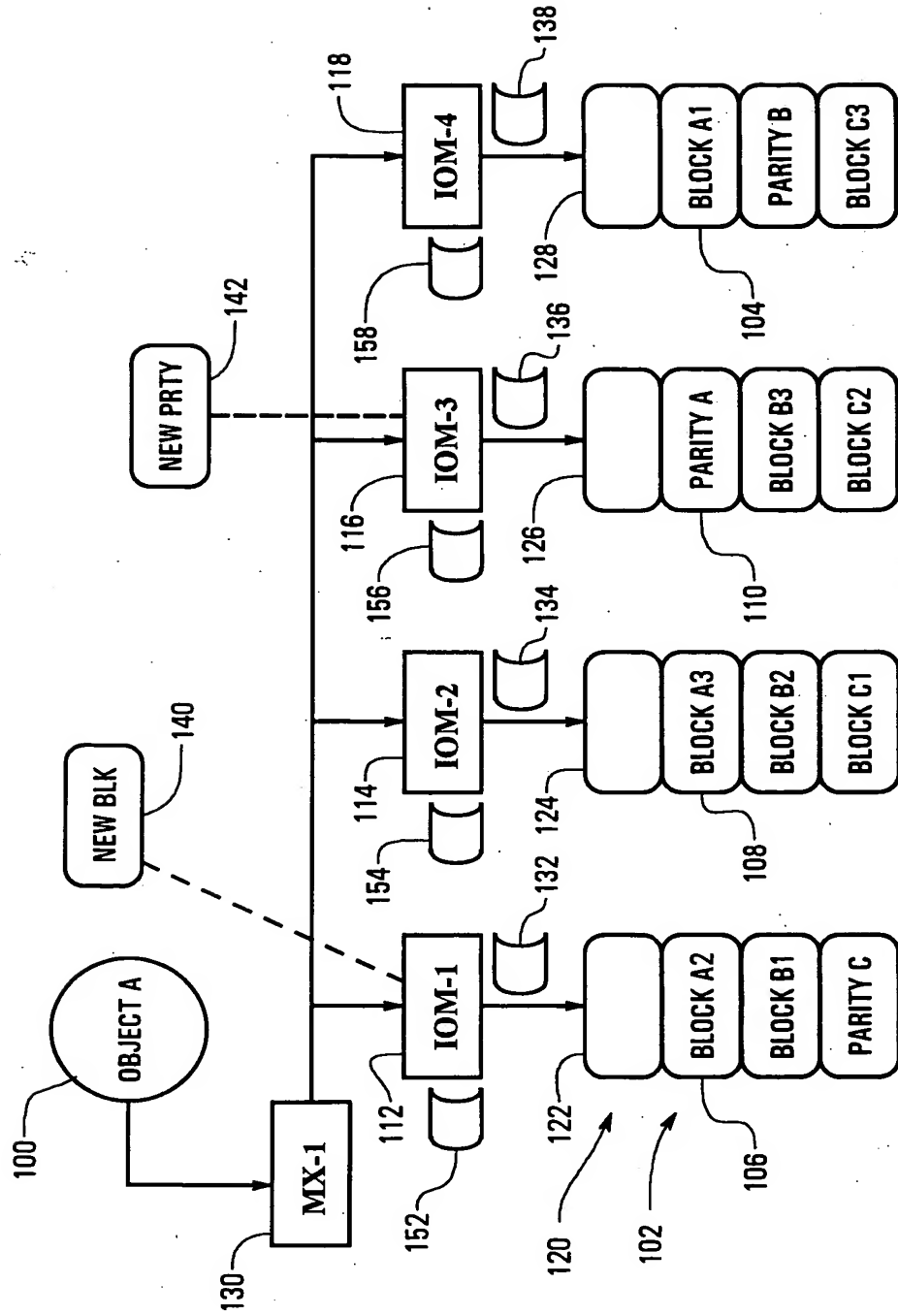
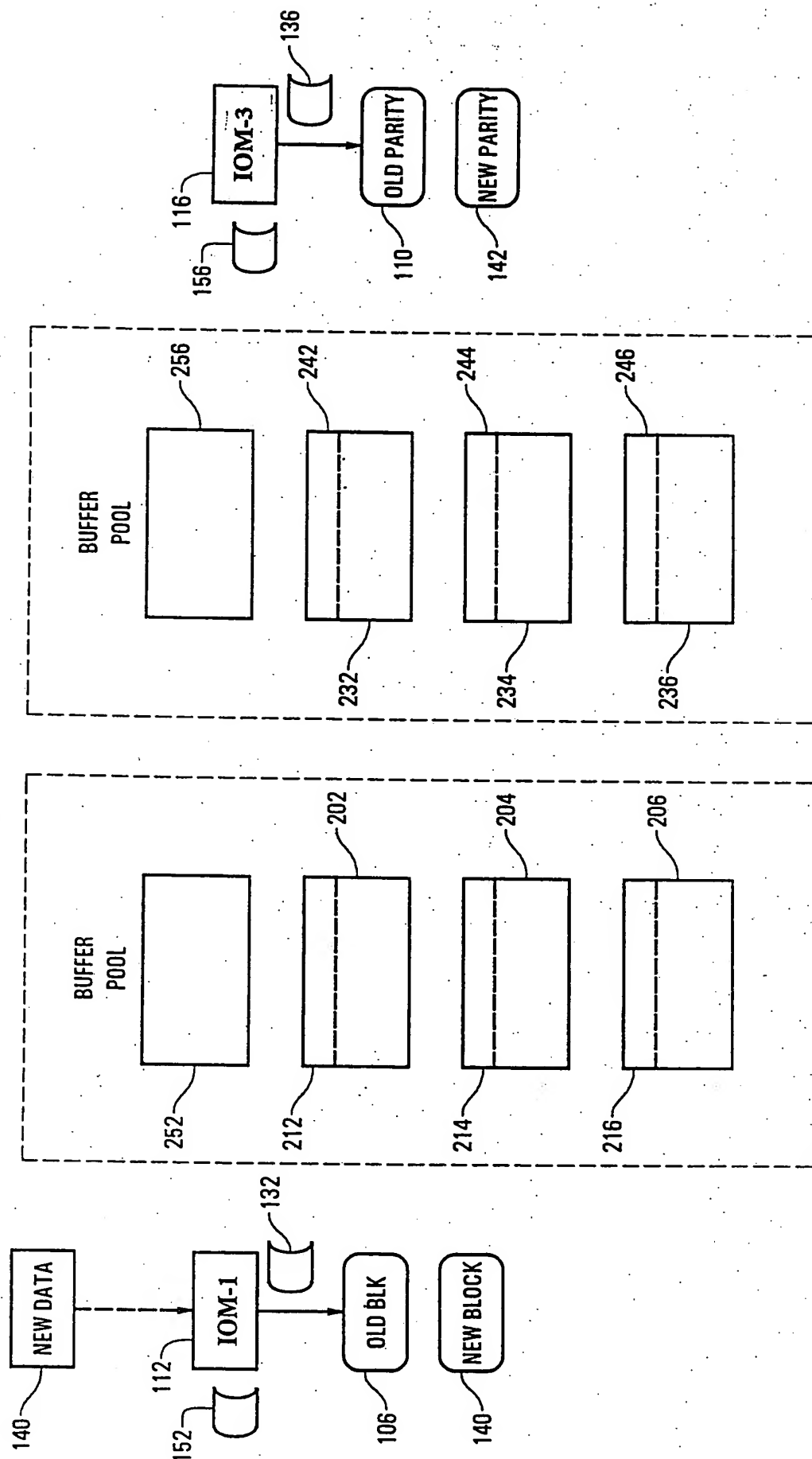


Fig. 2



**Fig. 3**

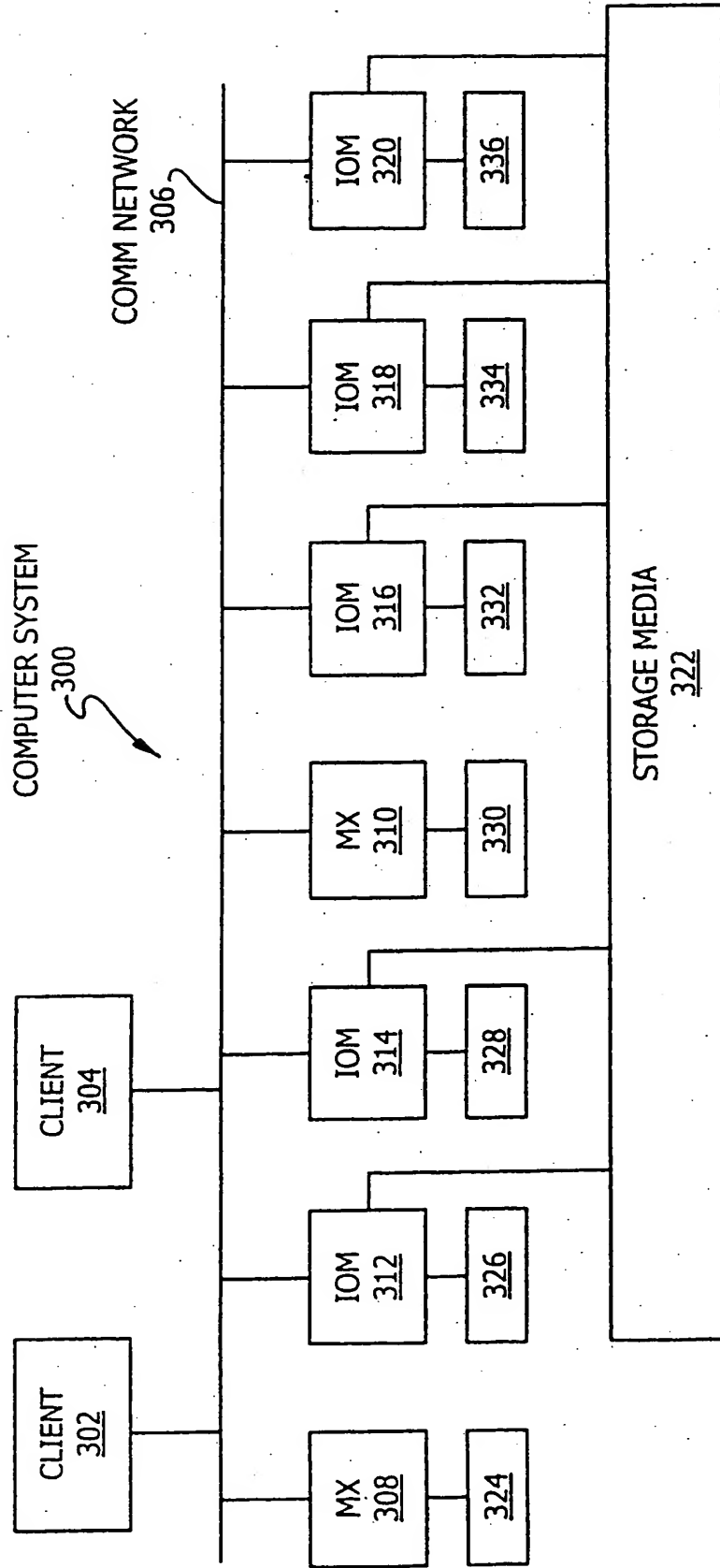
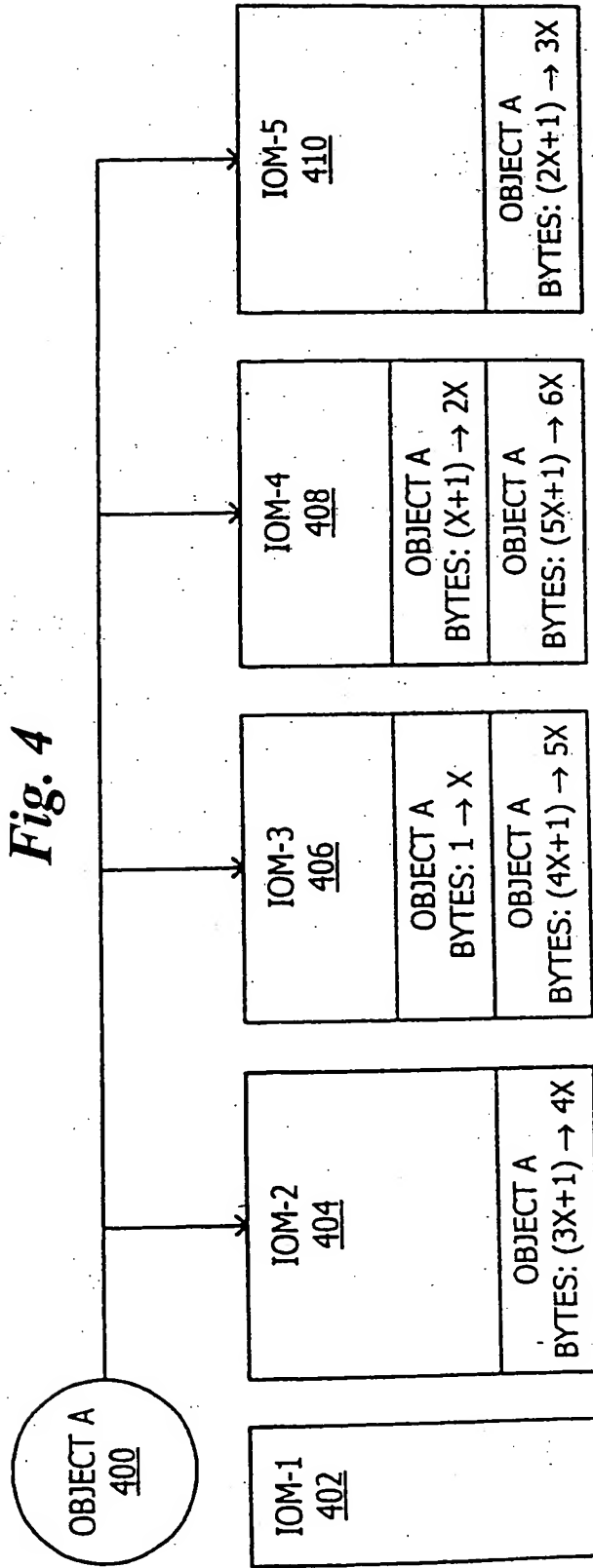
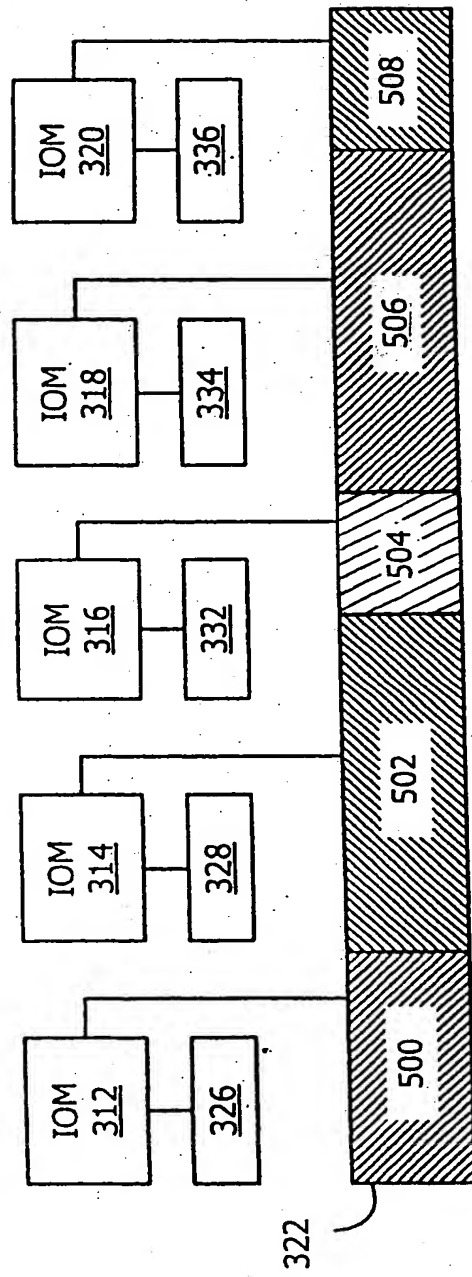


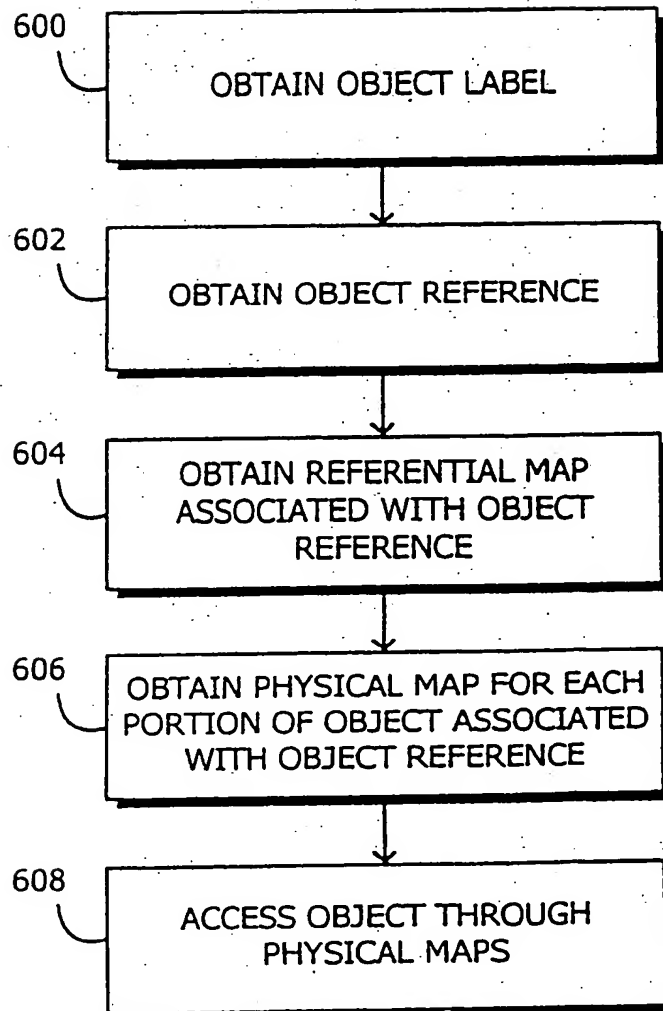
Fig. 4



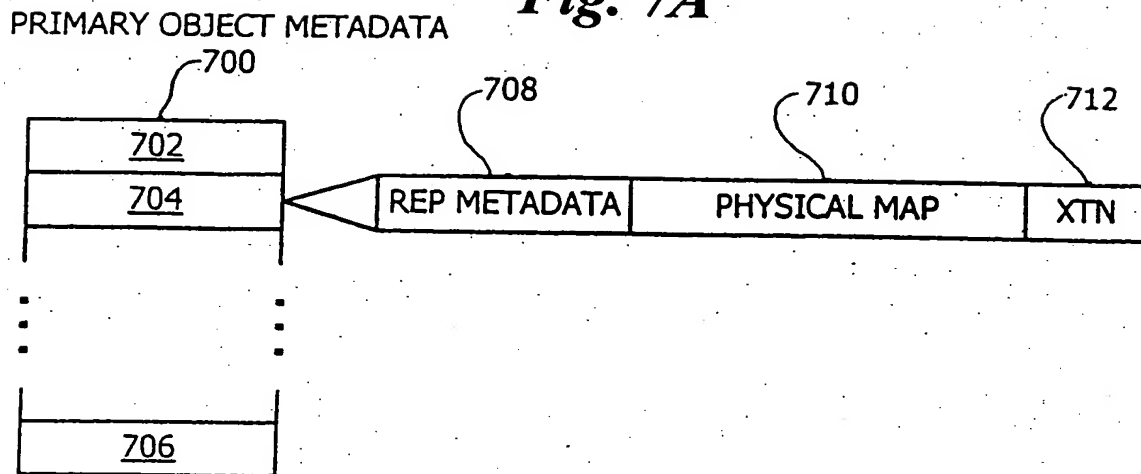
*Fig. 5*



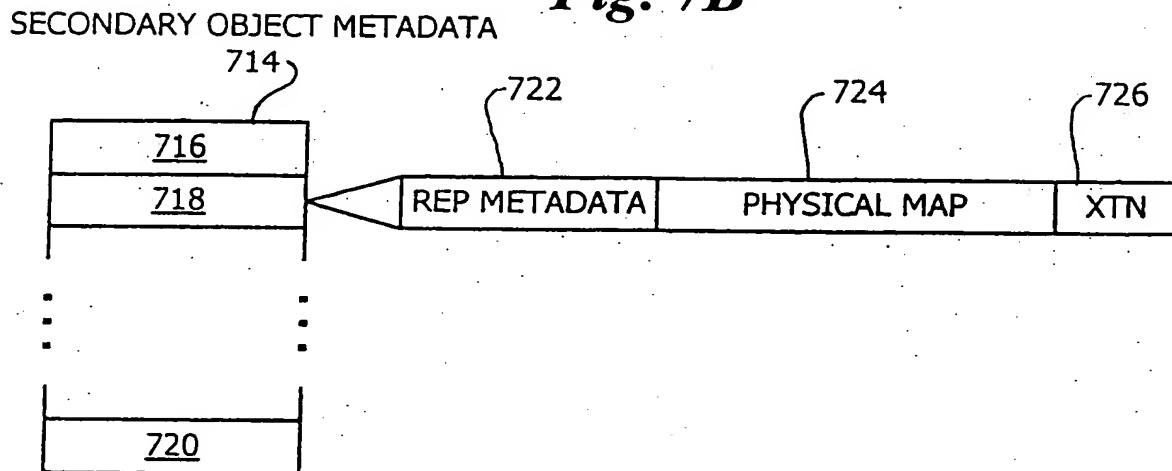
***Fig. 6***



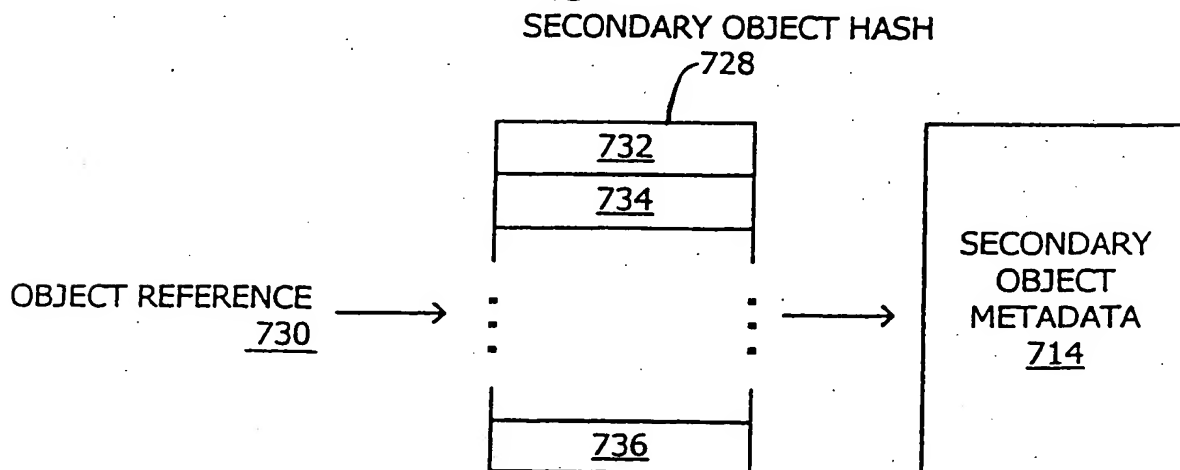
**Fig. 7A**



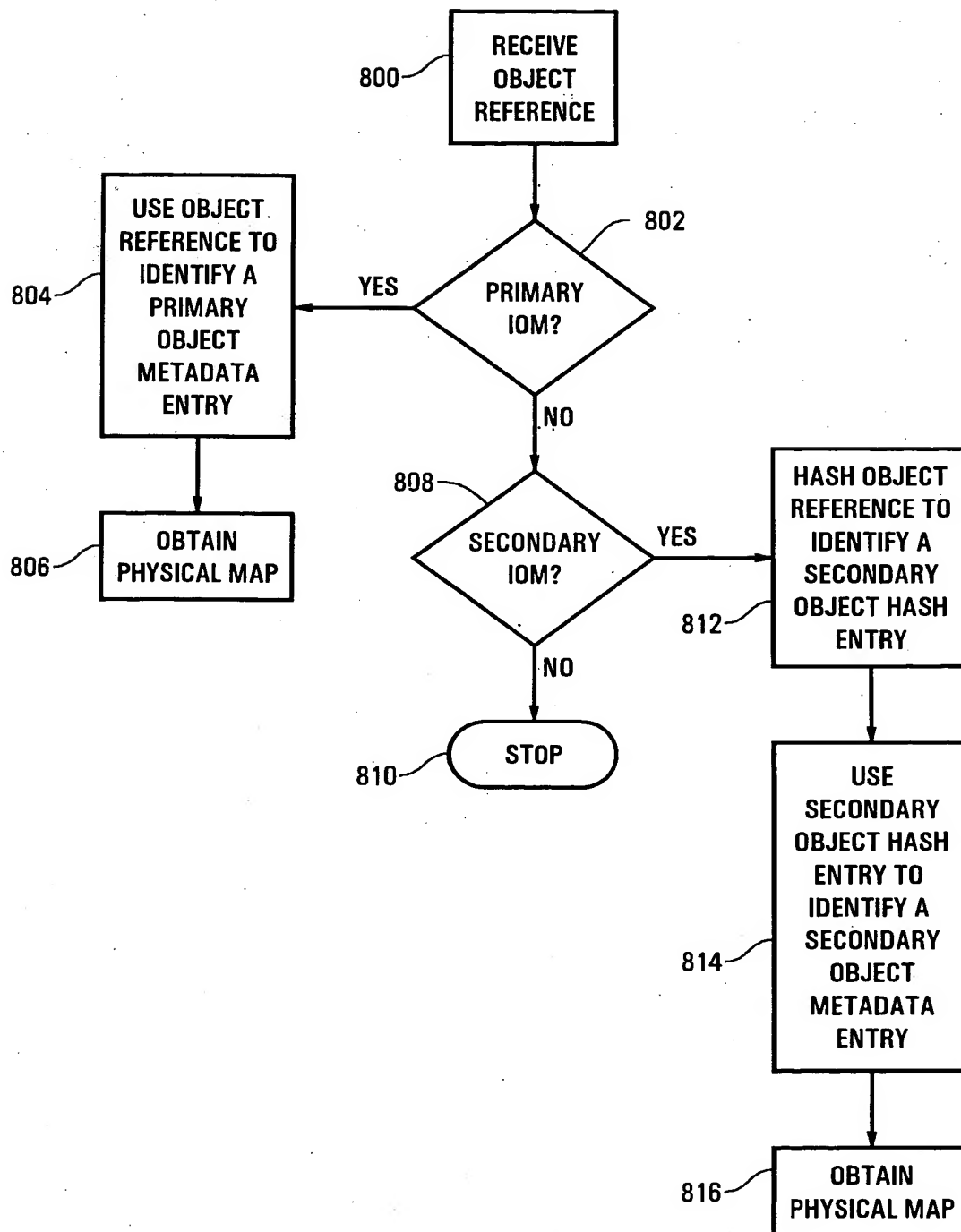
**Fig. 7B**



**Fig. 7C**



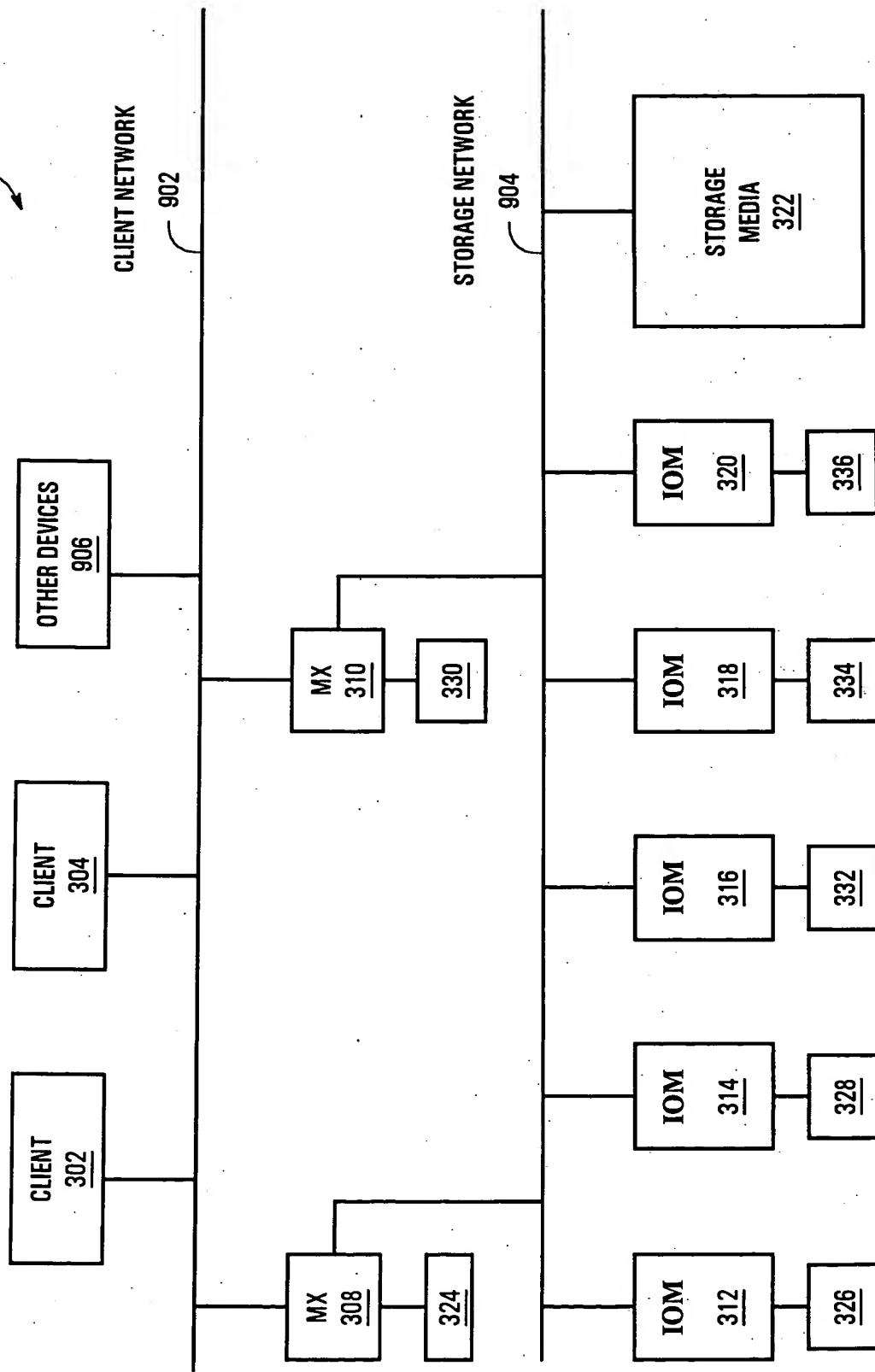
**Fig. 8**





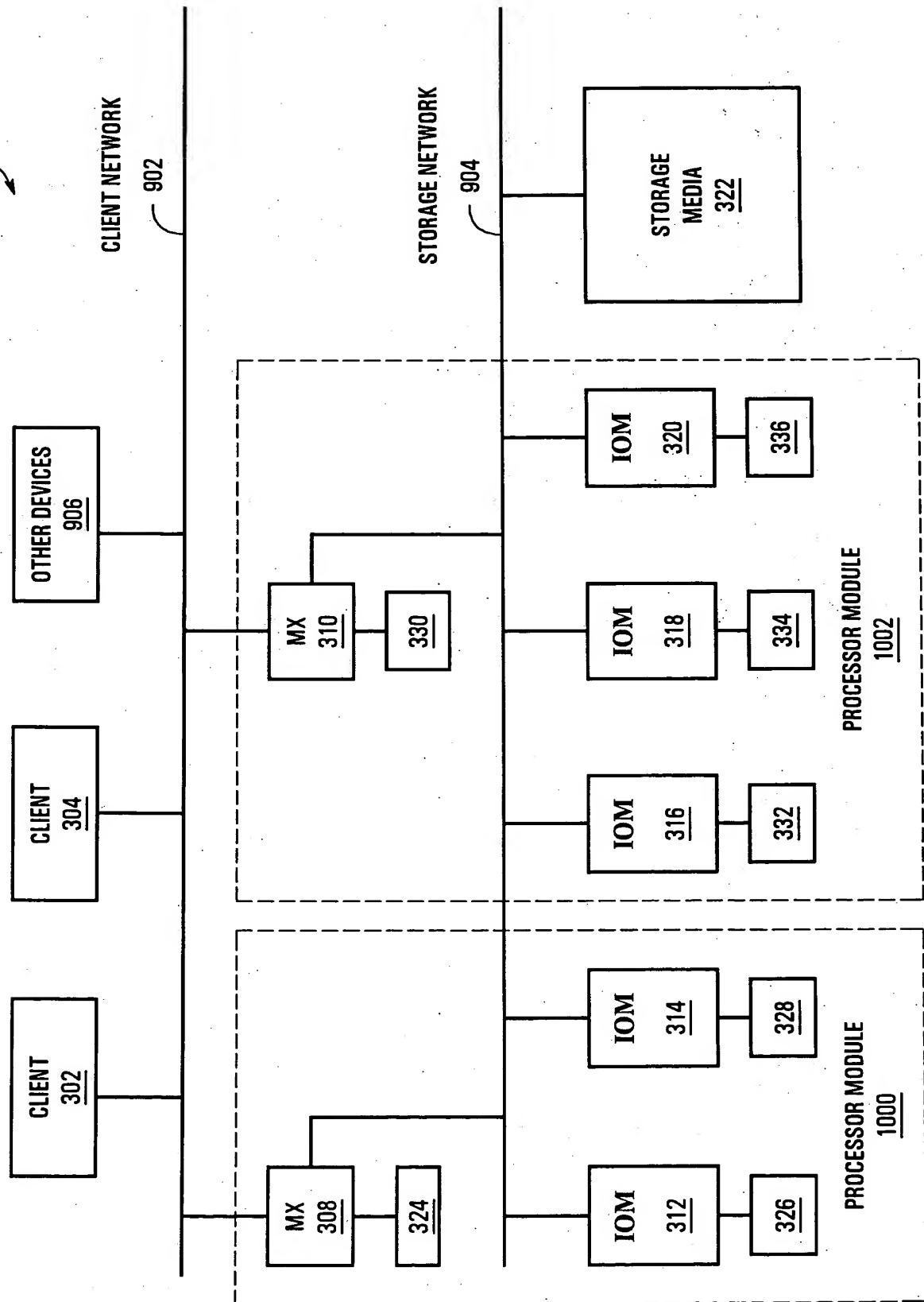
COMPUTER SYSTEM  
900

Fig. 9

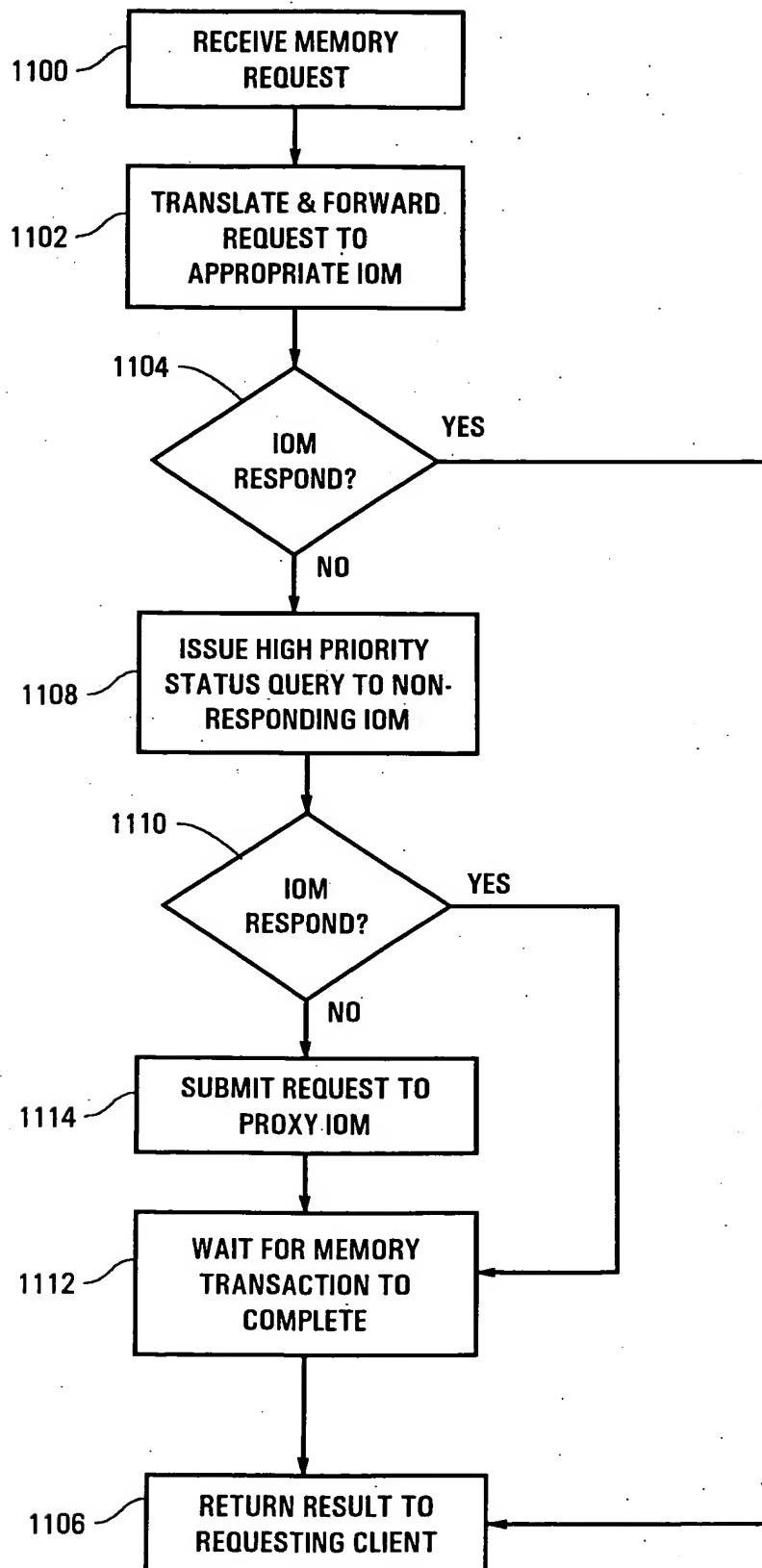


**Fig. 10**

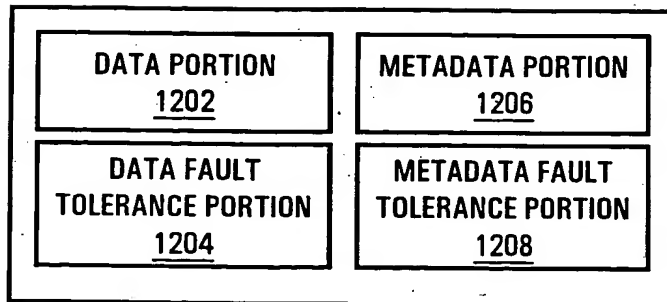
COMPUTER SYSTEM  
900



**Fig. 11**



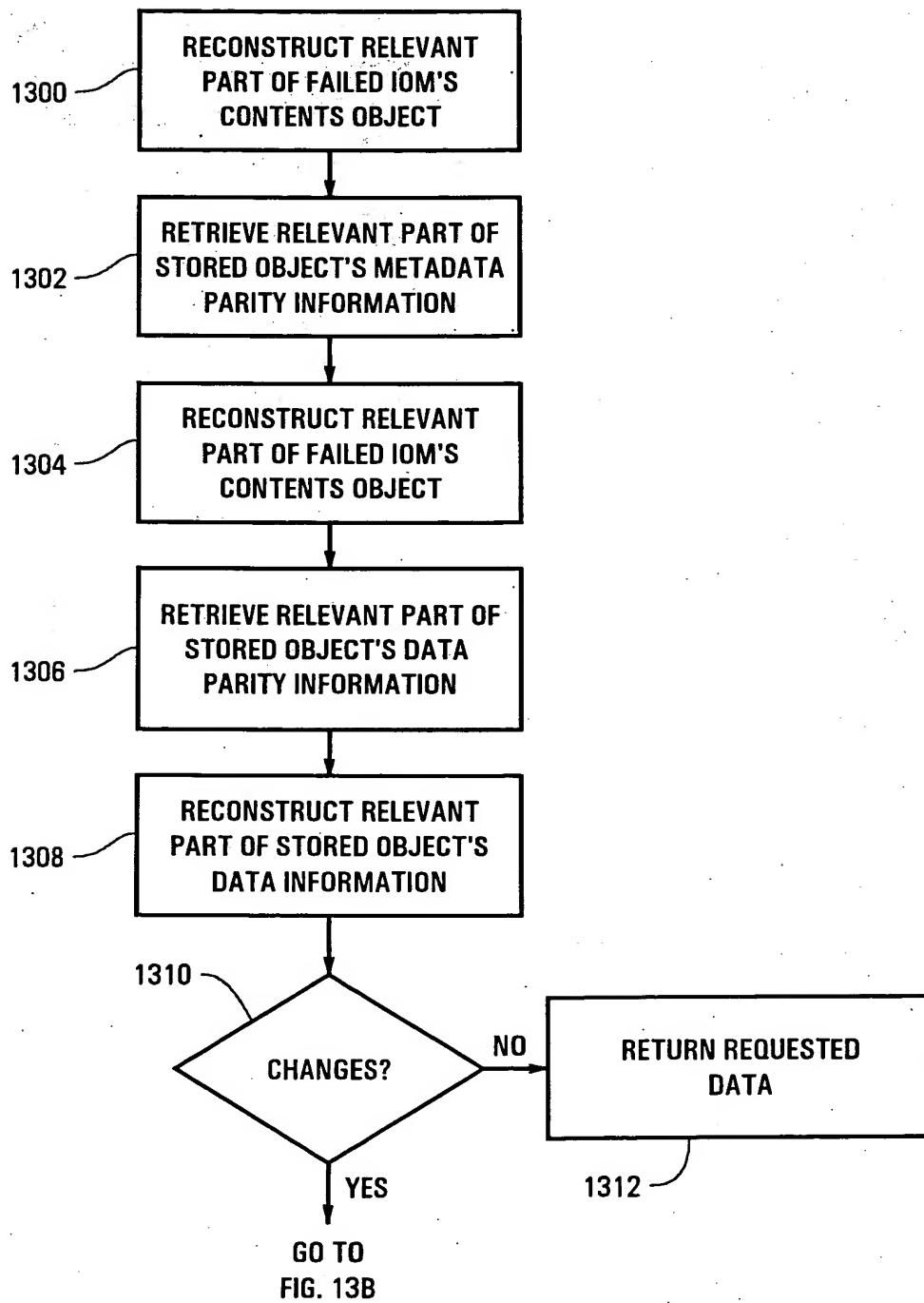
***Fig. 12***



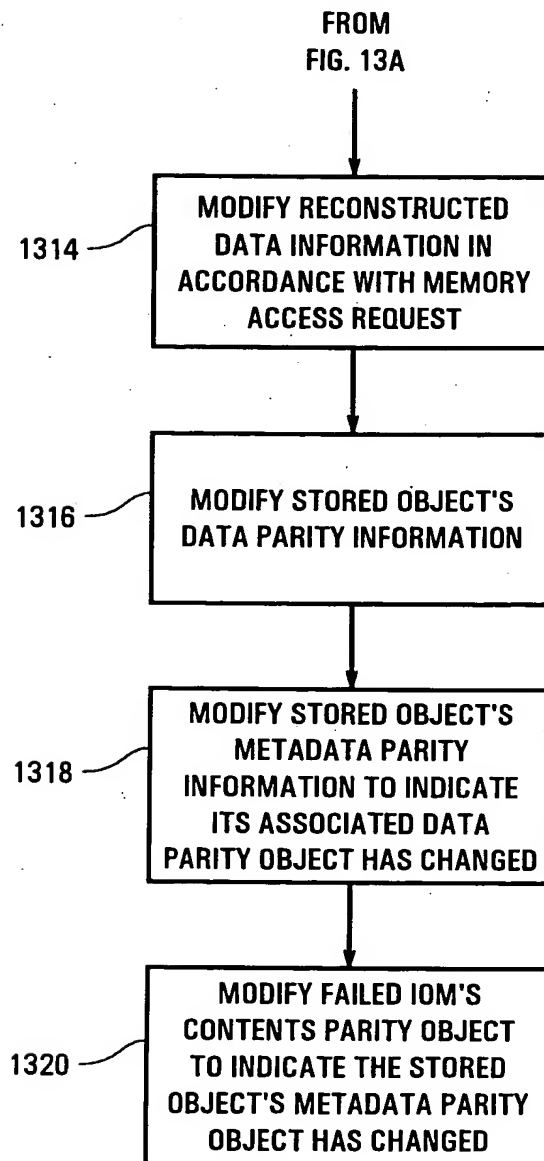
**STORED OBJECT**

1200

**Fig. 13A**



**Fig. 13B**



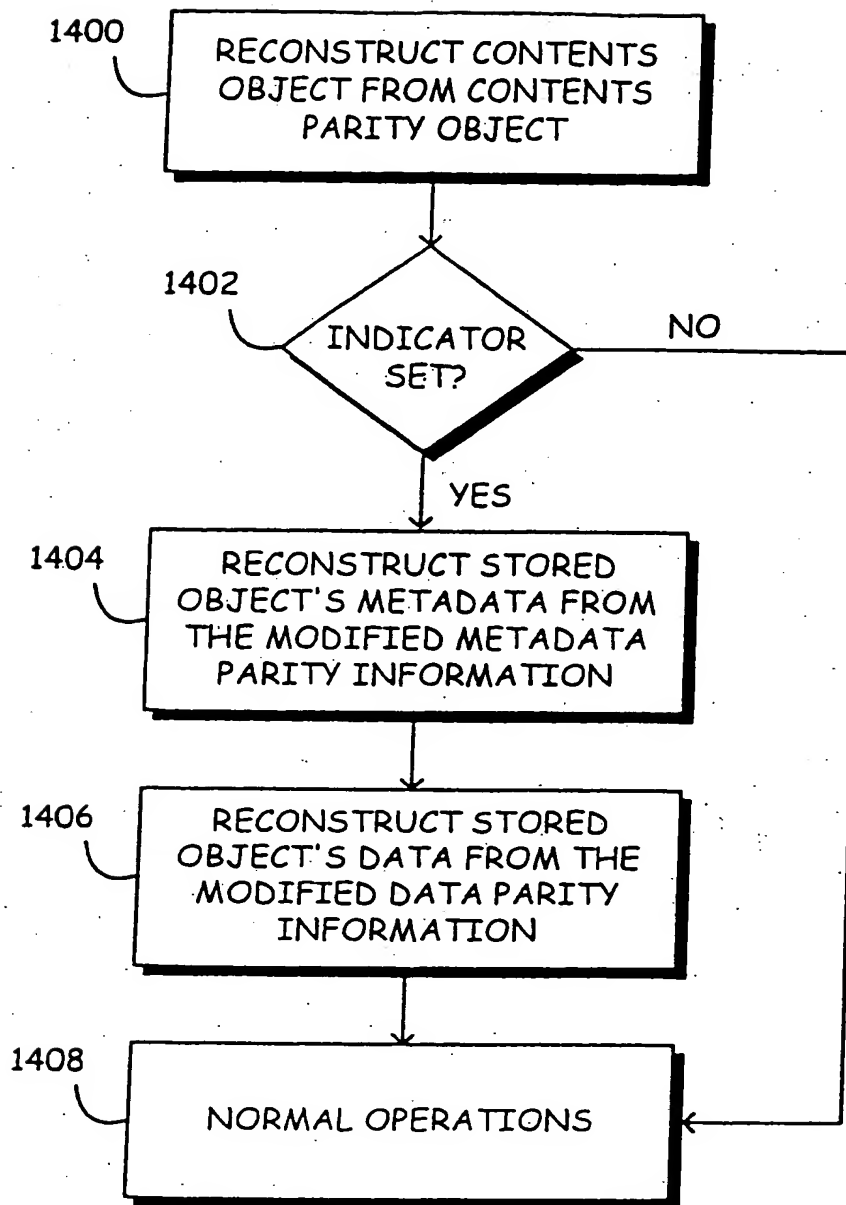


FIG. 14

***Fig. 15***

***Fig. 15A***

***Fig. 15B***



**Fig. 15A**

Time-line for a Data Write and Its Parity Update		
At the Data IOM	Msg	At the Parity IOM
Read old data with flags = DP_DONTBLOCK    PF_BC_READAHEAD    PF_BC_CREATE.		
Call modifyBlock getting a dirty buffer to hold new data transferred from caller (with a new disk address if needed).		
Make a journal entry for the write with a new ID = mj and the GFID, lblk, and new and old disk addresses for data.		
Transfer data into buffer. Set write_start and write_bytes to indicate the range of dirty bytes in the buffer that must be parity updated.		
Mark buffer tough and write data to disk with flags = DP_DONTBLOCK.		
Reply to caller if Toughness 0.		
Send ios_updatepn request to parity node with write_start, write_bytes, journal (mj & contents).	→	Receive ios_updatepn request.
Merge new data with old-data if necessary.		Get buffer for parity delta or data.
Compute parity delta or data (in a new buffer if necessary).		Write a new journal entry, pj, containing parity GFID, lblk, new and old disk addresses, and

**Fig. 15B**

			data of mj (i.e., mj, GFID, lblk, new and old disk address) with flags = DP_DONTBLOCK.
Receive request for parity delta or data, or the parity journal data.	←		Request new parity delta or data.
Return the parity delta. Then release the old data/delta buffer.	→		If needed, get old-parity with flags = DP_DONTBLOCK    PF_BC_READAHEAD
Reply to caller if Toughness 1.			Receive the parity delta in the new parity buffer.
When new data buffer is on disk, mark new data disk address in the cnode or indirect. Mark mj in their and the bitmap's pf_jmrange.newest.			Merge the parity delta or data with the old parity buffer if needed. Then write new parity to disk with flags = DP_DONTBLOCK.
When mj and the new data buffer are on disk, send data hardened message to the parity node.	→		When new parity buffer is on disk, mark the new disk address and pj in pf_jmrange.newest of the parity cnode or indirect buffer and of the bitmap.
Receive the hardened message from the parity node. If error, send update_pn to parity proxy.	←		Send parity update hardened message to the data node.
Reply to caller if Toughness 2.			Receive the hardened msg from the data node.
Deallocate the old data disk address if any and journal the mj_complete entry.			Finally, deallocate the old parity disk address if any and journal the pj_complete entry.

***Fig. 16A***

NO OLD DATA CHUNKS



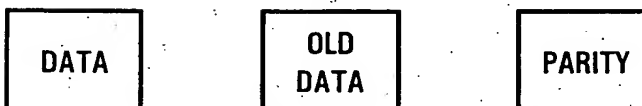
***Fig. 16B***

PARTY HAS OLD CHUNK



***Fig. 16C***

DATA HAS OLD CHUNK



***Fig. 16D***

BOTH PARITY AND DATA HAVE OLD CHUNKS

